A Solaris Backup Script How-To

Stanley J. Hearn
GIAC GSEC Practical v 1.3
February 3, 2002

Abstract

A backup strategy is more complex then creating a redundant copy of disk storage and considering the strategy a success. A successful backup strategy must detail how the backup media are rotated, how the media are archived, how the system will be recovered, and what the backup software will do to create the backup. Although all parts of the backup strategy are equally important, this paper will focus on the backup script and will detail a flexible backup script that uses built-in Solaris software tools which create a reliable local backup of a Solaris machine running Oracle.

Introduction

The backup script will accomplish the following goals:

- Create a backup archive that is as easy to restore a single file as it is to restore an entire file system.
- The backup script will run autonomously. The only human intervention will be to swap media and review output.
- All mounted file systems will be backed up, not just a fixed list. When new file systems are mounted, they are automatically backed up without having to modify the script.
- Allow for a list of file systems to be excluded from back up.
- Meet the backup requirements of Oracle.
  - Place all tablespaces in backup state, not just a fixed list. When new tablespaces are added, they are automatically dealt with without having to modify the script.
  - separately backup Oracle redo logs
- The script will create a detailed log of the backup.
- The script will send an abbreviated email summary of the backup to the administrator
- After a successful backup, the script will verify to some extent the contents of the backup media.
- Check the system log for hardware errors.
- The backup script will be able to run on any Solaris 2.6 or greater machine without modification.

What follows is a description of a Bourne shell backup script designed to run on Solaris 2.6 or greater running Oracle that will meet all of the above criteria. See Listing 1.

Before We Begin

It is essential that any backup script be tested before being placed into production. A complete backup procedure would also require periodic testing to ensure that it continues to meet the backup requirements including the ability to successfully recover from a catastrophic system failure in a timely manner.

Remember to meet the backup requirements of all non operating system applications running on the system. Recovering to the point that the operating system is running is merely the first part of the restoration process. The second and equally important part is application data recovery. If the application can't be recovered, then the disaster still exists.

Print disk partition information and any hardware-specific information and place it in a safe secure location. This information will make a disaster recovery less painful. You can use this script to print the partition information for disks on a Solaris system. I cannot stress enough the importance of having this information at hand when you're in crisis mode during a disaster.

```
format < /dev/null | \
nawk '/[0-9].*\./{system ("prtvtoc /dev/rdsk/" $2 "s2") ; \
print "\n\n\n------"}' > $TEMP_FILE
```

Don’t ever buy more disk space than you have the ability to back up. When you calculate the cost of purchasing more disk space, always include the cost of enough tape storage to accommodate it. The backup script described in this paper requires that all data fit on one tape for unattended operation.

The Backup Script's Nuts and Bolts

All backup tools have their strengths and weaknesses. This script will use two tools to copy information to our media, ufsdump and cpio. Each one has advantages over the other. The script will leverage the advantages of both by creating backups that alternate between them.

ufsdump and ufsrestore are the Solaris equivalent of dump and restore. In Solaris the default file system is UFS. Dump and restore are dependent on the underlying file system type. You must restore a dump to the same type of file system as its source. A dump from different system types, Linux, Irix, or Solaris are not compatible. ufsdump creates a backup of an entire disk partition. It can perform a full or incremental backup. However, this backup script will always perform a full level 0 dump. The ufsrestore tool has an interactive restore feature of browsing the tape file image and selecting exactly which files to retrieve.

cpio has the ability to back up individual files. cpio is also capable of backing up all mounted file systems in one archive. However, we will back up each file system separately. The reason for doing this is that it makes it easier and quicker to restore if you have many small file systems stored on the media instead of one very large one. A system that won't boot and only needs the root file system restored will be more convenient to recover if our root partition is in a separate cpio archive.

The cpio version that ships with Solaris doesn't have an option to restore an absolute cpio archive to a relative location. Therefore we will only create cpio relative archives. If you're disaster recovery plan allows you to use non-standard operating system tools, then I would suggest using GNU cpio, but make sure you have a compiled copy that can be easily restored to a failed system booted to CDROM recovery mode. You could also overcome this limitation of absolute restoration by chroot-ing to the recovery mount point and performing the restore.

The script will create a relative cpio archive, but how relative? If you list the contents of a cpio archive it is not always obvious what mount point the archive was generated from. It may be easy to determine if its a standard mount point. For example, if you see directories like `./joe`, `./steve`, `./stan`, etc, it would be obvious that this archive was generated from `/home`. But if you saw directories like `./ORACLE`, `./serverroots`, etc, it would be less obvious which file system they belong in. On an Oracle system the `./ORACLE` directory can in fact appear at several mount points.

Because of this ambiguity the backup script performs a cpio archive with file names relative to the root directory. This simply means that instead of backing up a file system relative to itself or its mount point, it will be backed up relative to root. For example a file system named `src` in `/usr/local/src`, won't be backed up as `./`, but will be backed up with the name of `./usr/local/src`. Don't be confused, we are still backing up individual file systems, we're just making sure that their file names will be relative to the root directory.

However, Solaris cpio has a problem with archiving the root directory relative to itself. Therefore we will place a sed statement directly following the find command to convert `"./"` to `"/"`.

```
find .${FS} -print -xdev | \
sed -e 's/^\.\/$/\//' | \
cpio -oacvC 32768 –O /dev/rmt/0 \
2>>/usr/local/cpio-log/write.020101.err \
1>>/usr/local/cpio-log/write.020101
```

You are probably quite familiar with the options used for cpio. The `-O /dev/rmt/0` allows for the separation of the archive device, standard error, and standard output. We can log the list of files we've sent to cpio in one log file and log errors in another.

Special Considerations must be taken to back up Oracle databases. Simply backing up the file systems that contain the data files is not enough. These files are very large and change continuously. There are several methods to back up Oracle databases, and more than one method should be implemented. Three methods are described below.

The first method is a cold backup. This involves completely shutting down the database which restricts any process or user from modifying the data files. The data files become static and are simply copied like any other unchanging file. Cold backups on a production system are not practical because during the backup the database is unavailable.

The second method involves exporting the database information to disk files and is known as a database export. Database exports can be done while the database is running making it a better alternative than cold backups for production systems. Also by exporting your database to an NFS mounted file system, this will further safeguard your database for recovery purposes. Just make sure that the NFS host machine is backed up adequately and is secure. The export method is also flexible in that one can pick and choose what to restore from an export. For example, if someone accidentally drops a database table, just that table can be retrieved from the last export. Run the export function in a cron job daily at a time when the database has the least amount of activity.

The third method, which the backup script will directly utilize, is called a hot or on-line backup. This involves running the database in the ARCHIVELOG mode. When the tablespaces are placed in backup mode, the main data files will remain quiescent and all database transactions will be written to the redo logs. This gives the backup program time to backup these large data files while they are not changing. After these files are copied, the database is taken out of ARCHIVELOG mode and the redo logs are copied to the backup media. CAUTION: Do not begin a database on-line backup during peak database usage times because Oracle will write complete database blocks instead of just transactions to the redo logs and this can create excessive database logging. NOTE: if the script abnormally exits, you'll need to manually take the tablespaces out of backup mode.

To restore a database that was backed up using the ARCHIVELOG mode method, you would restore the datafiles and the redo logs. Then you would play the redo logs forward to bring the database exactly back to the same state when the redo logs were created. Consult with your database administrator to verify that you're backing up everything he will need to recover from a disaster and practice your disaster recovery.

Before the backup begins, put each tablespace in backup mode. After the backup is complete take each tablespace out of backup mode. Then back up the redo logs which contain the database transactions. When the script is backing up an Oracle system, the redo logs are always added as a cpio archive at the end of all the file system backups on the media. Even if a ufsdump backup is performed, there will always be a cpio archive at the end of the tape for systems running Oracle.

The backup script will automatically use Oracle's server manager to obtain a list of current tablespace names and write them to a file. It will do this automatically so that any new table spaces that are added will always get backed up without requiring script modifications. Then it will insert commands before and after the tablespace names to create two Oracle server manager scripts, one script to put the tablespace names in backup mode and one to take them out of backup mode.

These are the Oracle server manager commands to prepare a tablespace for backup and to end the backup.

```
     alter tablespace [tablespace_name] begin backup  # before
     alter tablespace [tablespace_name] end backup    # after
```

At the end of the backup when the tablespaces are taken out of backup state, the script sleeps for 120 seconds to allow for all transactions to be flushed to the redo logs. Then the redo logs are written to the tape in a cpio archive.

It will help the database administrator perform a replay of the redo logs after a disaster if he has the archive sequence numbers. The backup script will execute an `archive log list` server manager command to get this information for inclusion in our backup report.

This script only places one Oracle database instance in ARCHIVELOG mode. It would not be too difficult to modify the script to allow for multiple instances.

Magnetic Tape Control (mt) sends commands to a tape drive. Many of these commands are familiar, but some are not. The script will use these mt commands.

- rewind – rewind the tape
- rewoffl – rewind the tape and eject it (go offline)
- eom – space to end of recorded media on tape
- weof – write count EOF marks at current position on tape
- status – display current status of tape

The summary file is the first thing that goes on the tape. This summary file will serve to identify this backup tape. When you need to identify a tape or restore a system, you'll appreciate this summary file. The summary will contain:

- The name of the system that created the backup.
- The date when the backup was created.
- The format of the backup. Was it created with cpio or dump?
- A list of file systems, their mount points and sizes (output of df command).
- The order in which the file systems appear on the tape.
- Recommended procedures for restoring the tape.

The summary file can be read from the tape using the command:

```
cat < /dev/rmt/0
```

A backup configuration file is used to keep machine specific configurations so that the administrator can make one copy of the backup script for all Solaris systems and place host specific values outside of the backup script.

The host configurable parameters are:

`TAPEDEV` is the tape device name with rewind
`TAPEDEV_NR` is the tape device name with norewind
`HARDWARE_LOG` is the name of the system log
`BU_LOG_DIR` is the directory where the backup logs are created
`EXCLUDE_FS` space delimited set of file systems to not include in the backup
`IS_ORACLE_MACHINE` Is this a machine running Oracle?
`ORACLE_REDO_LOG_DIR` is the directory where the redo logs are kept
`ORACLE_HOME home` directory of Oracle
`ORACLE_SID` instance SID of Oracle database to backup

The Backup Script Flow (Listing 1)

The script will keep a count of the number of file marks that are written to the tape. At the end of the backup we'll use that count to validate the number of file marks actually on the tape.

One argument to our backup script will determine between a cpio and ufsdump backup. It will assume that if anything besides "dump" is specified, a cpio backup will be performed.

The script will set a few default values for configuration variables and then source the configuration file.

The script will verify that the variables TAPEDEV and TAPEDEV_NR point to character special devices. If they do not, it will exit with an error message.

When the system is writing to tape, tape error messages will be written to the system log. If you don't examine the log, you'll not know when there are "sense key media errors" or "excessive write errors." You may have a problem and not know about it. The backup script should display the entries that are put into the log during the backup process. This can be done by counting the number of lines in the log when the backup began.

```
     LOG_START=`wc -l < /var/adm/messages`
```

The summary file is created and copied to the tape. This is the first file that appears on all backup tapes.

When the backup is finished, display the entries from this count to the end of the log file to show any hardware errors that may have been placed there.

```
     cat /var/adm/messages | sed -e "1,${LOG_START}d"
```

Be careful to not run backups at a time when the system logs are being rotated. The output will not be accurate.

Attempt to rewind the tape device. If this fails, exit with an error message. Also write an eof mark to the tape to make sure it is writable.

```
     mt –f /dev/rmt/0 weof 1
```

The return status of mt is checked by using this if statement after a rewind attempt and after writing an weof.

```
     if [ ${?} –ne 0 ]; then
       echo "Error Message"
       exit 1
     fi
```

Programmatically get the list of currently mounted file systems. On Solaris the file systems are called UFS file systems. The –F option for the df command specifies which file system type to list. We then pipe the output to cut to get only the first column of the df output. Finally we sort the file systems so that dependent file systems get backed up before independent ones (`/usr/local` is backed up before `/usr/local/src`, etc) This is done to facilitate a restore.

```
     FS=`df -F ufs | cut -d' ' -f1 | sort`
```

Because we are automatically obtaining a list of mounted file systems to back up, we have included a facility to exclude file systems from the backup. The variable EXCLUDE_FS may contain a space-delimited list of file systems for this purpose. The script steps through a list of each mounted file system to see if it appears in our excluded list. If it doesn't appear, then it adds that file system to the list of file systems to back up. The variable FILESYSTEMS will contain the list of the file systems the script will back up.

The next task the script will perform is to get a list of the tablespaces and build the two Oracle server manager scripts we'll use during the backup. We'll then put the tablespaces in backup mode.

Depending on the parameter passed to the backup script, it will perform a ufsdump or a cpio back up of the file systems.

If the backup is of an Oracle machine, then the Oracle tablespaces are taken out of backup mode and the redo logs are written as a cpio archive at the end of the tape.

After the tape is created, we'll count the number of file systems on the tape and compare that to the number that we wrote to the tape and print a message indicating our findings. The `eom` command will move the tape forward just past the last archive file. The `status` command will display how many file marks are on the tape.

```
     /usr/bin/mt -f /dev/rmt/0n eom
     FMARKS_COUNT=`/usr/bin/mt -f /dev/rmt/0n status | \
      awk '/file no/ { print $3 }'`
```

When we're done we'll output the date and time and then we'll rewind the tape and eject it. It is important to eject the tape to safeguard against overwriting the backup.

Scheduling the Backup

Schedule the backup job when the system and database have the least amount of activity. The output of the script should be emailed to the administrator. An example crontab entry that would run at 1:15 am would look like the following. It schedules a ufsdump backup to occur on Monday, Wednesday and Friday. A cpio backup will occur on Tuesday and Thursday.

```
15 1 * * 1,3,5   /usr/local/bin/backup dump 2>&1 | \
   /usr/ucb/Mail –s "dump Backup" sysadmin@yourdoman.com
15 1 * * 2,4   /usr/local/bin/backup 2>&1 | \
   /usr/ucb/Mail –s "cpio Backup" sysadmin@yourdoman.com
```

Conclusion

This paper demonstrates that a reliable backup script can be created with built-in UNIX tools. The script satisfied all of the requirements set forth. The script created a backup that is easy to restore by making all tape archives relative to the root directory and by placing a helpful backup summary description at the beginning of each backup tape. A host-specific configuration file will allow this script to be used on many Solaris machines without modification. It was written to run without human intervention. The script programmatically obtains a list of mounted file systems to back up.

Special considerations were made to accommodate the Oracle database application software by placing the tablespaces in backup state using the Oracle server manager before backing them up and by separately backing up the Oracle redo logs. All output from the backup tools was logged and an abbreviated output was generated which included any errors written to the systems error log. After the backup was complete the script verified that the file archive count that existed on the tape was equal to the file archive count written to the tape.

There are a few areas in which this script could be improved

- allow for multiple tape devices
- allow for backups to remote tape drives
- allow for additional command line parameters to specify things like alternate configuration file, included or excluded file systems
- deal with multiple Oracle database instances
- perform more extensive integrity checks of backup data
- migrate to Kourne shell for using subroutines

```
#!/bin/sh
#
#
# Usage: backup [ cpio | dump ]
#
#   cpio   Perform backup using cpio (default)
#   dump   Perform backup using ufsdump
#

# Set default configuration variables
TAPEDEV='/dev/rmt/0c'
TAPEDEV_NR='/dev/rmt/0cn'
HARDWARE_LOG='/var/adm/messages'
BU_LOG_DIR='/usr/local/backuplogs'
EXCLUDE_FS=""

export TAPEDEV TAPEDEV_NR HARDWARE_LOG BU_LOG_DIR EXCLUDE_FS

IS_ORACLE_MACHINE="NO" # Machine doesn't have Oracle installed
ORACLE_REDO_LOG_DIR=""
ORACLE_HOME=""
ORACLE_SID=""
export IS_ORACLE_MACHINE
export ORACLE_REDO_LOG_DIR ORACLE_HOME ORACLE_SID

CFG='/etc/backup.conf'

# See if host configuration file exists
# if so then source it
if [ -f $CFG ]; then
# Get different settings from default
  . $CFG
fi

# Get date in the format YYMMDD
YMD=`date +%y%m%d`

if [ ! -c ${TAPEDEV} ]; then
  echo "${TAPEDEV} is not a character special device"
  exit 1
elif [ ! -c ${TAPEDEV_NR} ]; then
  echo "${TAPEDEV_NR} is not a character special device"
  exit 1
fi

if [ ! -d ${BU_LOG_DIR} -o ! -w ${BU_LOG_DIR} ]; then
  echo "${BU_LOG_DIR} doesn't exist or isn't writable"
  exit 1
fi

echo "### Backup Starting at `date`"

# LOG_START is the number of lines in our hardware log
# when the backup began.  We're not concerned with anything
# that occurred before now.
LOG_START=`wc -l < ${HARDWARE_LOG}`
# strip leading and trailing spaces
```

```
LOG_START=`expr ${LOG_START}`

# Rewind tape
mt -f ${TAPEDEV} rewind

# Check mt return status
if [ ${?} -ne 0 ]; then
    echo "*** No tape loaded or drive offline ***"
    exit 1
fi

# Write EOF to tape to test that it is writable
mt -f ${TAPEDEV} weof 1

# Check mt return status
if [ ${?} -ne 0 ]; then
    echo "*** Tape is write protected ***"
    exit 1
fi

BACKUPTYPE="cpio"
if [ "${1}" = "dump" ]; then
  BACKUPTYPE="dump"
fi

# Get list of unix file systems to back up
MOUNTEDSYSTEMS=`df -F ufs | cut -d' ' -f1 | sort`

# Remove excluded file systems from list
for FS in $MOUNTEDSYSTEMS
do
  case " ${EXCLUDE_FS} " in

    # If its excluded do nothing
    *" ${FS} "*);;

    # Add entry to list
    *) case "${FILESYSTEMS}" in

        # List is empty so this is the first entry
        '') FILESYSTEMS="${FS}";;

        # List contains entries so append to list
        *) FILESYSTEMS="${FILESYSTEMS} ${FS}";;
    esac;;
  esac
done

# Build backup summary file to be put first on tape
cat << HEREIS > /tmp/backup.ident.$$

Backup of Hostname:`hostname` started on `date`

Tapedev: ${TAPEDEV}

${BACKUPTYPE} of:
```

```
### State of file systems to back up at start of backup
`df -k $FILESYSTEMS`
###

HEREIS

if [ -n "$EXCLUDE_FS" ]; then
  cat << HEREIS > /tmp/backup.ident.$$
### State of file systems to exclude from back up
`df -k $EXCLUDE_FS`
###

HEREIS
fi

if [ "${IS_ORACLE_MACHINE}" = "YES" ]; then
  cat << HEREIS >> /tmp/backup.ident.$$
###
  This machine is running Oracle.  The last archive on this tape
  Is a CPIO archive of the REDO logs in $ORACLE_REDO_LOG_DIR
###

HEREIS
fi
cat << HEREIS >> /tmp/backup.ident.$$
RESTORE INFORMATION:

All backups are done relative to their respective file systems.  Change to the
top level of the file system where the file should be restored before running
the restore command.

# cpio -icvC 32768 < /dev/rmt/0cn [FILE_TO_RESTORE]

# ufsrestore xvfs /dev/rmt/0 [FS_NO] [FILE_TO_RESTORE]
        NOTE: FS_NO is the number of file systems into the tape

HEREIS

# Write backup summary as first file on tape
cp /tmp/backup.ident.$$ ${TAPEDEV_NR}
rm /tmp/backup.ident.$$

# Keep count of number of file archives written to tape
# start at 1 because the backup summary
FS_COUNT=1

if [ "${IS_ORACLE_MACHINE}" = "YES" ]; then
  ######
  # Oracle Specific Section
  ###

  # Will contain all tablespace names (Oracle adds .log extension)
  SN_FILE='/tmp/spacenames'

  # Invoke Oracle to get list of tablespace names
  echo
  su - oracle -c ${ORACLE_HOME}/bin/svrmgrl <<HEREIS
```

```
connect internal
set termout off
spool ${SN_FILE};
select tablespace_name from sys.dba_tablespaces;
spool off;
exit;
HEREIS

  # These are the Oracle sever manager script files that we will
  # create with our list of tablespace names
  DB_BEGIN="/tmp/oracle-backup-begin.${$}"
  DB_END="/tmp/oracle-backup-end.${$}"

  # Build begin backup and end backup ORACLE server manager scripts
  echo "connect internal" | tee ${DB_BEGIN} > ${DB_END}
  echo "archive log list" >> ${DB_BEGIN}

  # Remove header and trailing information from log and build tablespace
  # directives
  sed -e "1,2d" -e "\$d" ${SN_FILE}.log | \
  while read TABLESPACE
  do
    # Directive to begin backup state for the tablespace
    echo "alter tablespace ${TABLESPACE} begin backup;" >> ${DB_BEGIN}

    # Directive to end backup state for the tablespace
    echo "alter tablespace ${TABLESPACE} end backup;" >> ${DB_END}
  done

  # Finish up directives
  echo "archive log list" >> ${DB_END}
  echo "exit;" | tee -a ${DB_BEGIN} >> ${DB_END}

  # Set mode of db scripts
  chmod 600 ${DB_BEGIN} ${DB_END}

      # Put tablespaces into backup mode
      su - oracle -c ${ORACLE_HOME}/bin/svrmgrl < ${DB_BEGIN}
  #
  # End Oracle Specific
  ######
fi

cd /

if [ "$BACKUPTYPE" = "dump" ]; then
  # Backup using dump
  for FS in ${FILESYSTEMS}
  do
      echo "Dumping ${FS} ${TAPEDEV_NR}" | \
        tee -a ${BU_LOG_DIR}/bu.ufsdump.${YMD}

      /usr/sbin/ufsdump 0fu ${TAPEDEV_NR} ${FS} 2>&1 | \
        # we're only concerned with the last two lines of output
        # to determine if this was a successful dump
        tee -a ${BU_LOG_DIR}/bu.ufsdump.${YMD} | tail -3

      # Count dump of file system
      FS_COUNT=`expr ${FS_COUNT} + 1`
  done
else
  # Backup using cpio
  for FS in ${FILESYSTEMS}
  do
    echo "Writing cpio archive of ${FS} to ${TAPEDEV_NR}" |\
    tee -a ${BU_LOG_DIR}/bu.cpio.${YMD}.err \
    >> ${BU_LOG_DIR}/bu.cpio.${YMD}

    # The output of find will always be relative to root
    find .${FS} -print -xdev | \
      # Use sed to change "./" back to "/" so Solaris cpio won't bark
      sed -e 's/^\.\/$/\//' | \
      cpio -oacvC 32768 -O ${TAPEDEV_NR} \
      2>> ${BU_LOG_DIR}/bu.cpio.${YMD}.err \
      1>> ${BU_LOG_DIR}/bu.cpio.${YMD}

    # count cpio archive of file system
    FS_COUNT=`expr ${FS_COUNT} + 1`
  done
  cat ${BU_LOG_DIR}/bu.cpio.${YMD}.err
fi

if [ ${IS_ORACLE_MACHINE} = "YES" ]; then
  # Take tablespaces out of backup mode
  echo
  su - oracle -c ${ORACLE_HOME}/bin/svrmgrl < ${DB_END}

  # Wait for redo logs to be flushed
  sleep 120

  # Remove temp files
  rm ${SN_FILE}.log ${DB_BEGIN} ${DB_END}
  FS="${ORACLE_REDO_LOG_DIR}"
  cd $FS
  echo "Writing cpio archive of ${FS} to ${TAPEDEV_NR}" |\
  tee -a ${BU_LOG_DIR}/bu.cpio.redo.${YMD}.err >> ${BU_LOG_DIR}/bu.cpio.${YMD}

  find . -print -xdev -follow | \
   cpio -oacvC 32768 -O ${TAPEDEV_NR} \
   2>> ${BU_LOG_DIR}/bu.cpio.redo.${YMD}.err \
   1>> ${BU_LOG_DIR}/bu.cpio.${YMD}

  # count cpio archive of file system
  FS_COUNT=`expr ${FS_COUNT} + 1`
  cat ${BU_LOG_DIR}/bu.cpio.redo.${YMD}.err
fi

# Display filesystem information that was just backed up
echo "### Currently mounted file systems:"
df -k ${FILESYSTEMS}
echo "###"
```

```
LOG_END=`wc -l < ${HARDWARE_LOG}`
# strip leading and trailing spaces
LOG_END=`expr ${LOG_END}`
if [ "${LOG_BEGIN}" -ne "${LOG_END}" ]; then
  # Report errors in log during backup
  # add additional grep lines to remove normal messages appearing in your
  # system logging script
  echo "### System logs generated during backup:"
  cat ${HARDWARE_LOG} | \
    sed -e "1,${LOG_START}d" | \
    grep -v "connect from " | \
    grep –v "refused connection from " | \
    nawk '{ print substr($0,1,78) }; \
          length > 78 { print "                " substr ($0,79) }'
  echo "###"
fi

# Remove old log files
if [ "${BACKUPTYPE}" = "cpio" ]; then
  cd ${BU_LOG_DIR}
  find . -mtime +14 -follow -exec rm {} \;
  ls -lt ${BU_LOG_DIR}/*
  echo
fi

# Get number of file marks on tape
mt -f ${TAPEDEV_NR} eom
FMARKS_COUNT=`mt -f ${TAPEDEV_NR} status | awk '/file no/ { print $3 }'`

# Number of file systems and files on tape should match
if [ "${FMARKS_COUNT}" -ne "${FS_COUNT}" ]; then
  echo "*** ERROR: ${FS_COUNT} file systems written to tape. " \
  "${FMARKS_COUNT} actually on tape ***n"
else
  echo "${FS_COUNT} file systems written and ${FMARKS_COUNT} verified" \
  "on tape"
fi

# Eject tape
mt -f ${TAPEDEV} rewoffl

echo "Backup Finished at " `date`
```

References

Cooper, Clark. "Using One Script to Back Up Linux and Solaris." Sys Admin Magazine. April 2001 Volume 10 Number 04. URL: http://www.samag.com/documents/s=1152/sam0104b/ (13 Jan 2002)

Diamond, Ben. "Quick and Dirty Server/Workstation Replication Using ufsdump." Sys Admin Magazine. April 2001 Volume 10 Number 04. URL: http://www.samag.com/documents/s=1152/sam0104e/0104e.htm (13 Jan. 2002)

Enhanced Software Technologies. "11 Common Backup Mistakes and How To Avoid Them" eLinux: Linux Technology Solutions. © 2000 August 24, 2001. URL: http://www.elinux.com/articles/bru.jsp

Gebhards, Ghet. "Restoring the Sun." Sys Admin Magazine. June 1998, Vol. 7 Issue 6. URL: http://www.samag.com/documents/s=1190/sam9806a/9806a.htm (10 Jan. 2002)

Indiana University. "Administering Backups" (Unix Workstation System Administration Education Certification Course.) Indiana University Unix EdCert. October 7, 1997. URL: http://www.uwsg.iu.edu/edcert/session2/backup-overview.html

Naudé, Frank. "Oracle Backup and Recovery FAQ." Underground Oracle FAQ. January 21, 2002 Revision 2.02. URL: .http://www.orafaq.com/faqdbabr.htm (12 Dec 2001)

Preston, Curtis W. "Backup on a Budget." Sys Admin Magazine. July 2001 Volume 10 Number 07. URL: http://www.samag.com/documents/s=1148/sam0107o/ (2 Feb. 2002)

Preston, Curtis W. "Understanding Oracle Backup & Recovery." Sys Admin Magazine. September 2001 Volume 10 Number 9. URL: http://www.samag.com/documents/s=1146/sam0109g/0109g.htm (11 Nov. 2001)

Quantum Corporation. "Data Protections Best Practices." August 24, 2001. URL: http://www.dlttape.com/proveIt/steps/plan/best/ (3 Jan 2002)

Summit Software Design. "Backup and recovery options." September 9, 2000. URL: http://www.summitsoftwaredesign.com/articles/article6.html (3 Feb. 2002)

Sun Microsystems. "Backing Up and Restoring Data." (System Administration Guide Part I) Docs.Sun.Com. URL: http://docs.sun.com/ab2/coll.47.8/SYSADV1/@Ab2PageView/idmatch(BKUPCONCEPTS-57422)?DwebQuery=ufsdump&oqt=ufsdump&Ab2Lang=C&Ab2Enc=iso-8859-1#BKUPCONCEPTS-57422 (3 Feb. 2002)